

Programação Orientada a Objetos

Aula: 27/01/2022

Relacionamento entre Classes (Associação)

Turmas B e C

Profa. Nadia Félix

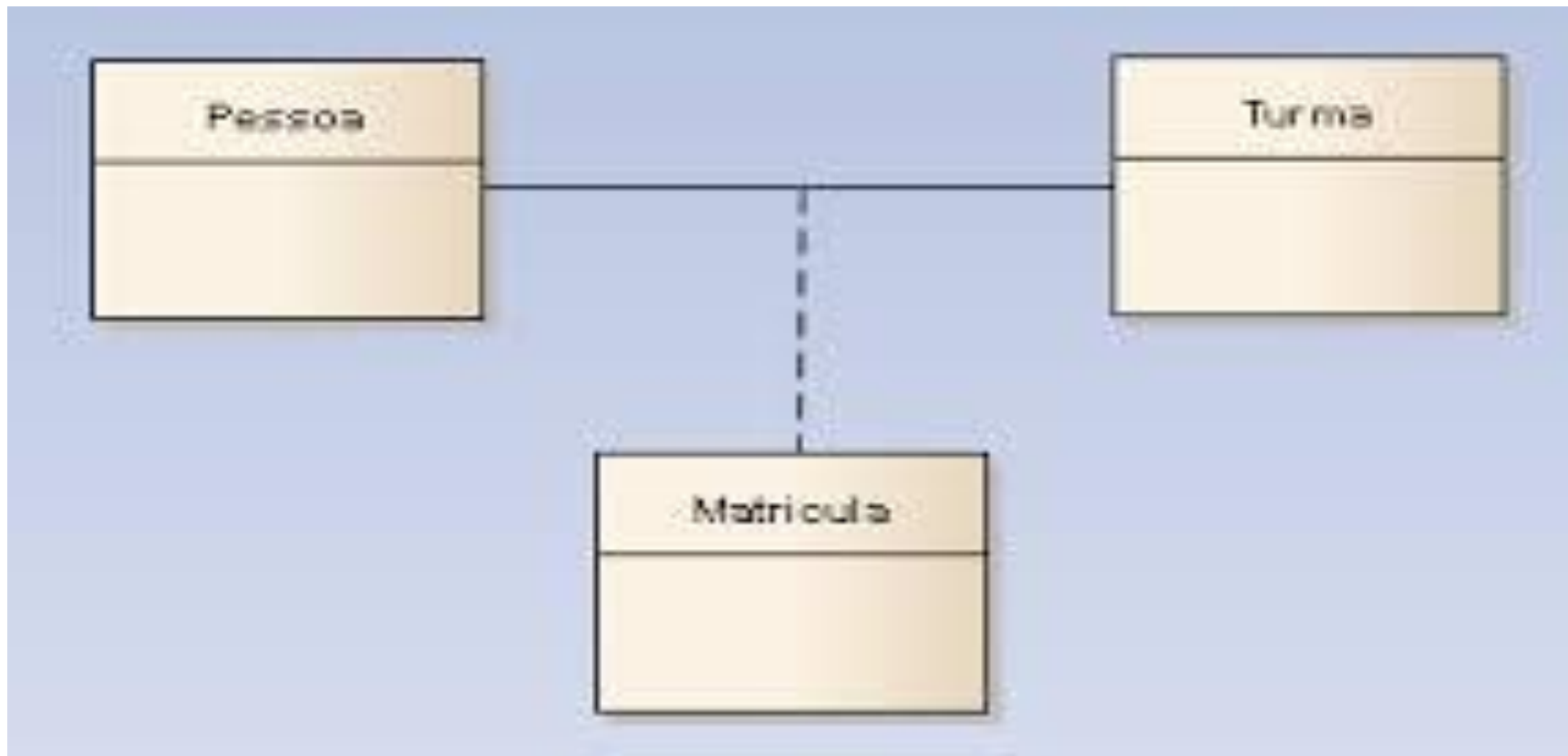
nadia.felix@ufg.br

Prof. Dirson Santos de Campos

dirson_campos@ufg.br

Programação Orientada a Objetos

RELACIONAMENTO ENTRE CLASSES (Associações)



Programação Orientada a Objetos

Relacionamento entre Classes

Em um sistema **Orientado a Objetos**, as classes não trabalham sozinhas, existem relacionamentos e comunicações entre elas.

O tipo do relacionamento e a forma de comunicação entre as classes e definem responsabilidades.

Programação Orientada a Objetos

Relacionamento entre Classes

Existem 3 tipos:

- Relacionamentos de Associação
(Agregação, Composição);
 - Agregação: estabelecem um vínculo entre objetos.
 - Composição: relacionamento do tipo todo/parte.
- Relacionamentos de Generalização (herança);
- Relacionamentos de Dependências

Programação Orientada a Objetos

- **Agregação**

Forma de composição em que o objeto composto apenas usa ou tem conhecimento da existência do(s) objeto(s) componente(s).

Os objetos componentes podem existir sem o agregado e vice-versa.

- **Composição**

Forma de associação em que o objeto composto é responsável pela existência dos componentes.

O componente não tem sentido fora da composição.

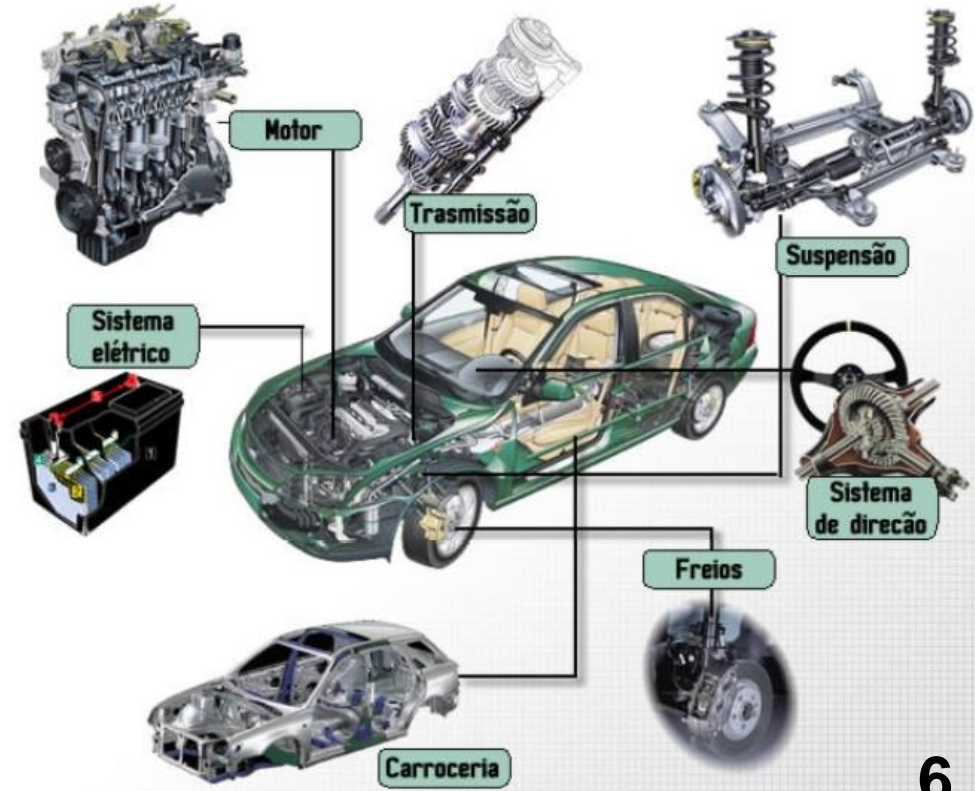
Programação Orientada a Objetos

Relacionamento entre Classes

Um automóvel, por exemplo, é formado por peças que apresentam relações muito definidas.

Possui: motor, rodas, banco, freio, Carroceria, volante e diversas outras engrenagens.

Esses elementos, associados ou conectados de maneira correta, permitem a existência e o correto funcionamento do automóvel.



Programação Orientada a Objetos

Relacionamento entre Classes

São alguns exemplos de relacionamento entre as peças que compõe um Automóvel:

- **Associação:** Pneus, Rodas, Amortecedores e Freios formam a Suspensão do Automóvel.
- **Generalização:** Filtro de Ar, Filtro de Óleo e Filtro de Combustível são tipos específicos de Filtro.
- **Dependência:** Motor depende de Bomba de Gasolina para bombear a gasolina para o seu interior.

Programação Orientada a Objetos

Relacionamento de Associação entre Classes em Java

- Java não possui uma forma declarativa para implementar agregações nem associações.
- Java apenas cria associações unidirecionais através de referências.

Exemplo:

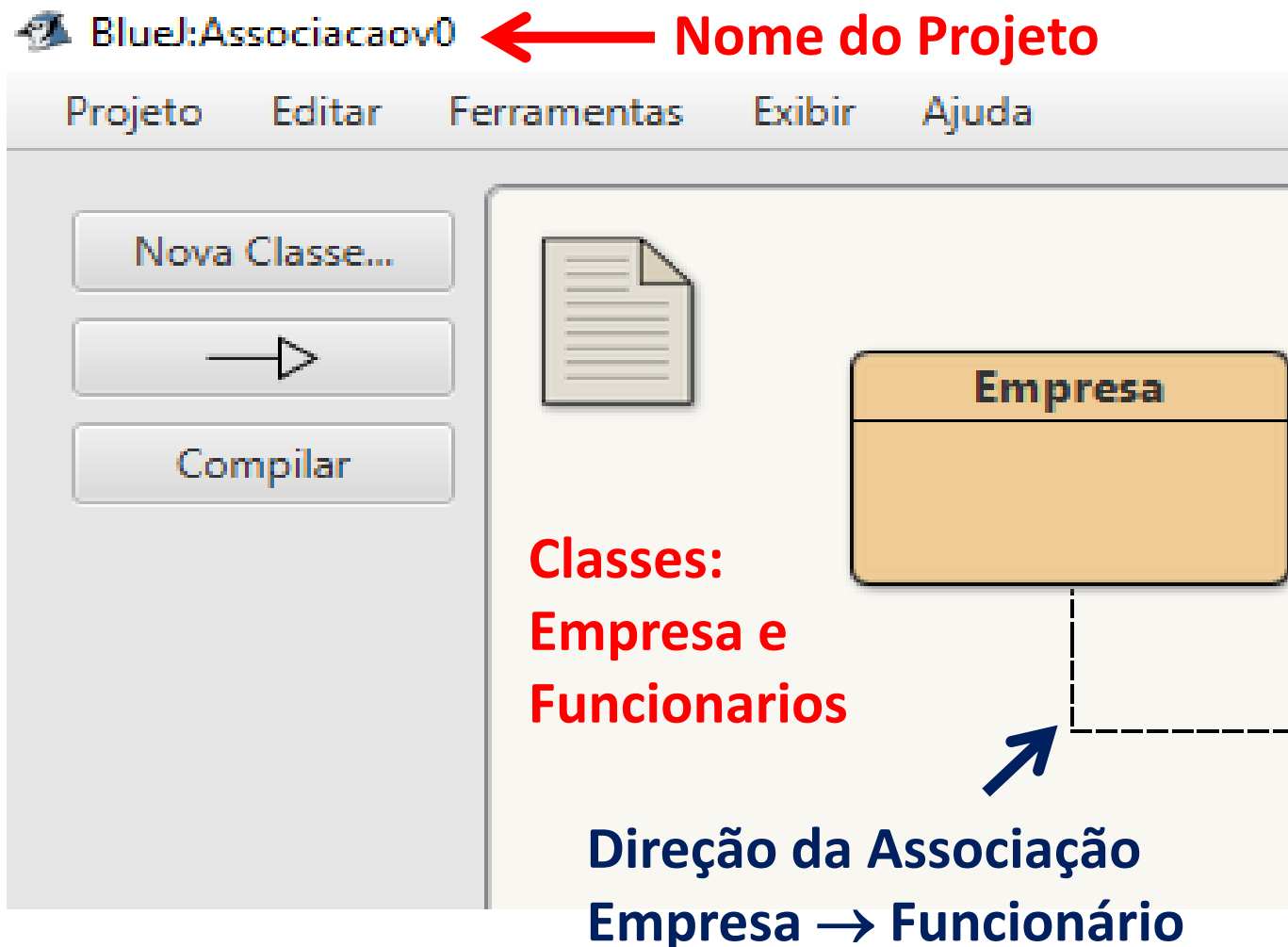
**Associação
unidirecional**

```
public class Conta {  
    private int codigo;  
    private Cliente cliente;  
    //...  
}
```


Programação Orientada a Objetos

Exemplo de Diagrama de Classes da UML adaptada pelo BlueJ

Visualização do Diagrama de Classes Empresa-Funcionário no BlueJ



O Diagrama de Classe é a forma de visualizar os relacionamento entre classes de um mesmo projeto adotado pelo BlueJ.

Programação Orientada a Objetos

- A **Direção de uma Associação**, representada pela direção da seta em um diagrama de classes, é chamada em POO de navegação.
 - Ela define qual classe tem acesso à outra classe.

Programação Orientada a Objetos

- Se a navegação da associação entre duas classes A e B for **A → B**, um objeto da classe A tem acesso a um ou mais objetos da classe B.
 - Isto significa que dentro de um objeto da classe A pode existir um ou mais objetos da classe B.
- Se a associação for **B → A**, a classe B é que terá um ou mais objetos de A.

Programação Orientada a Objetos

**Associando de Classes Empresa-Funcionário
com uso de encapsulamento**

**Recordando como acessar os atributos que
são privados (*private*) ?**

- Através de métodos de acesso

Padrão: GET e SET

Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário com encapsulamento

```
public class Empresa
```

```
{
```

```
    private String nome;
```

```
    private String cidade;
```

```
    private String fone;
```

```
    private String email;
```

```
    private Funcionario gerente;
```

```
}
```

```
public class Funcionario {
```

```
    private String nome;
```

```
    private String cargo;
```

```
    public void setNome(String nome) {
```

```
        this.nome = nome;
```

```
    }
```

```
    public void setCargo(String cargo) {
```

```
        this.cargo = cargo;
```

```
    }
```

```
    public String getNome() {
```

```
        return this.nome;
```

```
    }
```

```
    public String getCargo() {
```

```
        return this.cargo;
```

```
    }
```

```
}
```

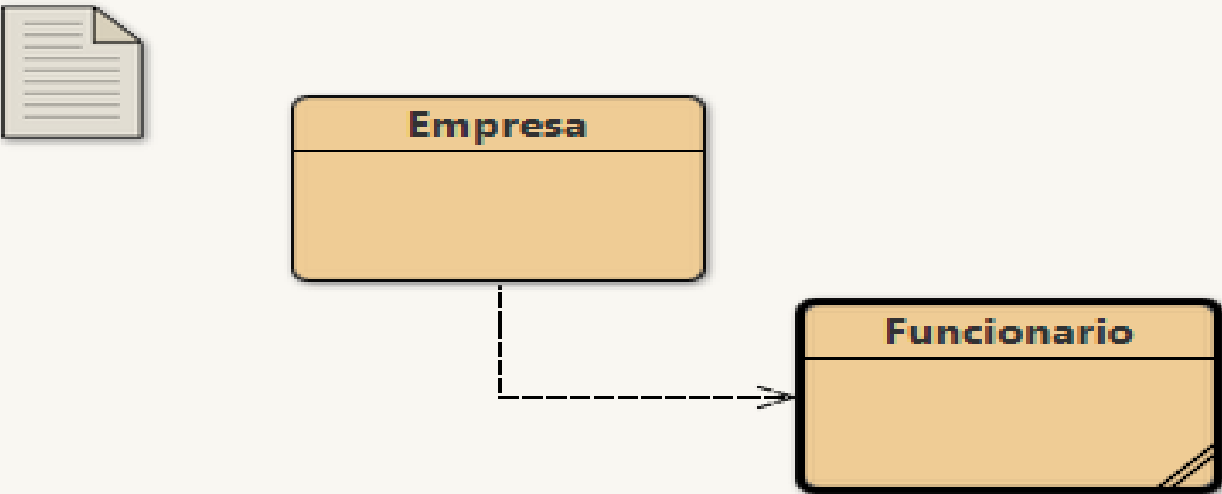
Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário

BlueJ:Associacaov0

Projeto Editar Ferramentas Exibir Ajuda

Nova Classe...
→
Compilar



```
classDiagram
    Empresa --> Funcionario
```

Selecione a Classe que deseja criar o objeto e depois o crie usando o botão direito do mouse.

empresa1:
Empresa func_1:
Funcionario

Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário

BlueJ:Associacao0

Projeto Editar Ferramentas Exibir Ajuda

Nova Classe...

→

Compilar

Empresa

Funcionario

empresa1: Empresa

func_1: Funcionario

BlueJ: Chamar Método

void setCargo(String cargo)

func_1.setCargo("Gerente")

OK Cancelar

```
classDiagram
    class Empresa
    class Funcionario
    Empresa --> Funcionario
```

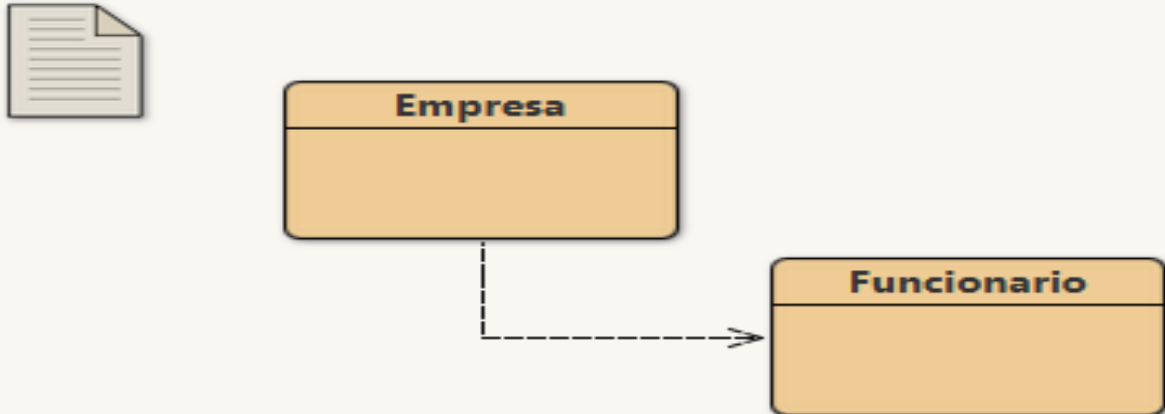
Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário

BlueJ: Associacao0

Projeto Editar Ferramentas Exibir Ajuda

Nova Classe...
→
Compilar



```
classDiagram
    class Empresa
    class Funcionario
    Empresa --> Funcionario
```

empresal: Empresa func_1: Funcionario

BlueJ: Chamar Método

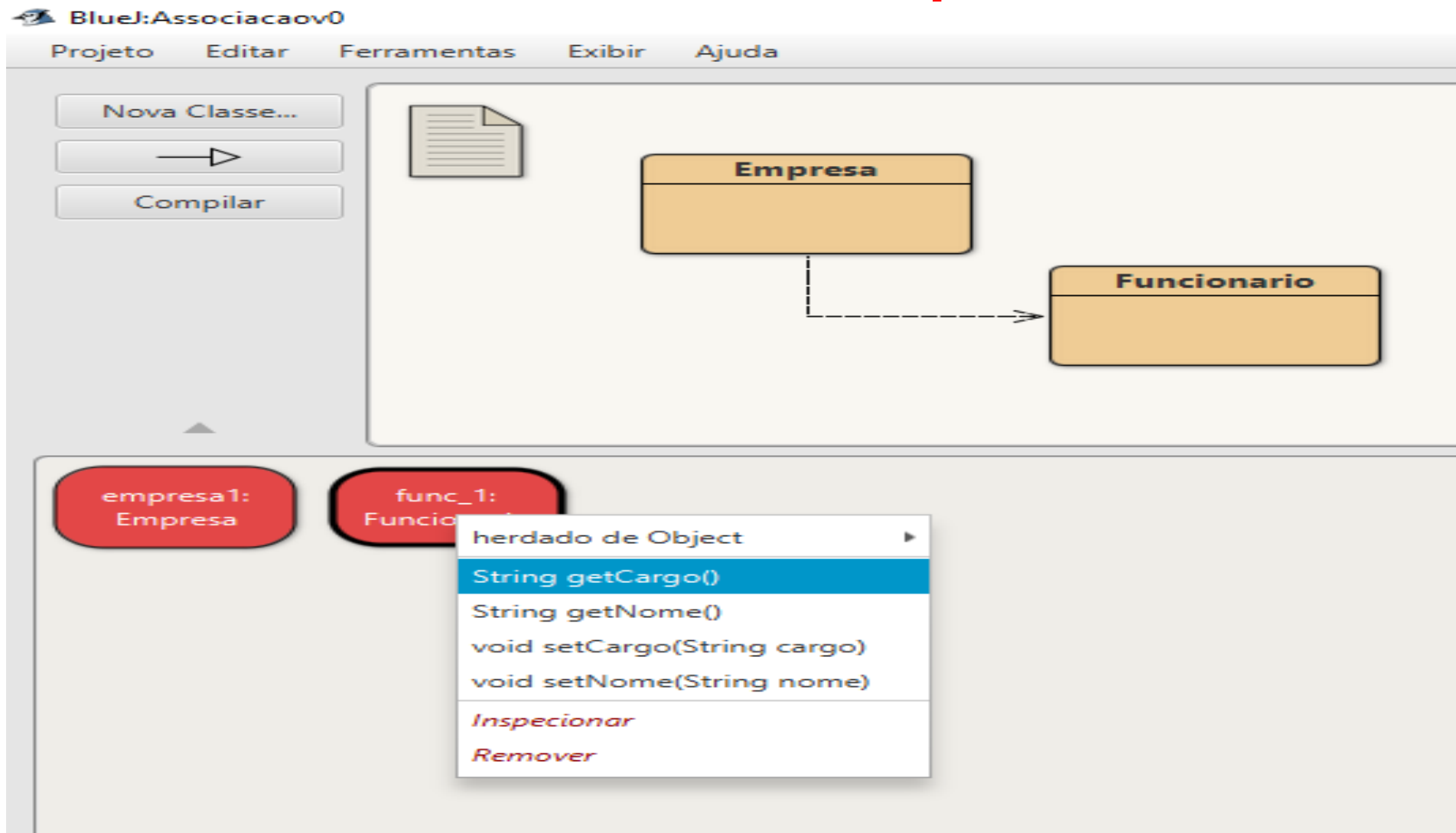
void setNome(String nome)

func_1.setNome("Fulano")

OK Cancelar

Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário



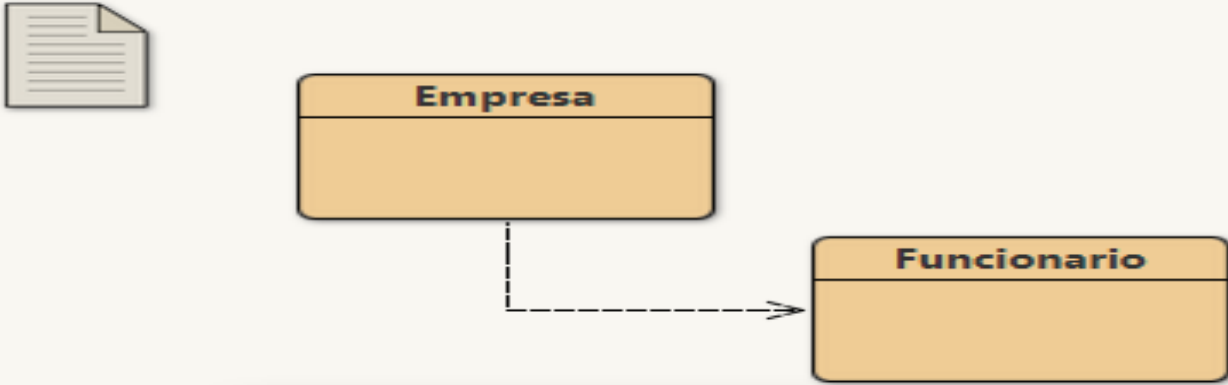
Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário

BlueJ: Associacao0

Projeto Editar Ferramentas Exibir Ajuda

Nova Classe...
→
Compilar



```
classDiagram
    class Empresa
    class Funcionario
    Empresa --> Funcionario
```

empres1: Empresa func_1: Funcionario

BlueJ: Resultado de método

String getCargo()

func_1.getCargo() retornado:

String	"Gerente"
--------	-----------

Inspecionar
Obter
Fechar

Programação Orientada a Objetos

Associando de Classes Empresa-Funcionário

The screenshot displays the BlueJ IDE interface. The main workspace shows two class boxes, **Empresa** and **Funcionario**, connected by a dashed association line. On the left, a sidebar contains buttons for **Nova Classe...**, a right-pointing arrow, and **Compilar**. Below these are two red buttons representing objects: **empresa1: Empresa** and **func_1: Funcionario**. A dialog window titled **BlueJ: Resultado de método** is open in the foreground, showing the result of the **func_1.getNome()** method call. The result is displayed as **String** followed by **"Fulano"**. The dialog includes buttons for **Inspecionar**, **Obter**, and **Fechar**.

Programação Orientada a Objetos

Modelo da Conta em UML

Conta
<ul style="list-style-type: none">~ numero : int~ saldo : double~ limite : double~ tipo : String
<ul style="list-style-type: none">+ sacar(valor : double) : void+ depositar(valor : double) : void+ transferir(valor : double, destino : Conta) : void

Programação Orientada a Objetos

Acrescentando Atributos à Conta

Informações necessárias para abrir uma conta:

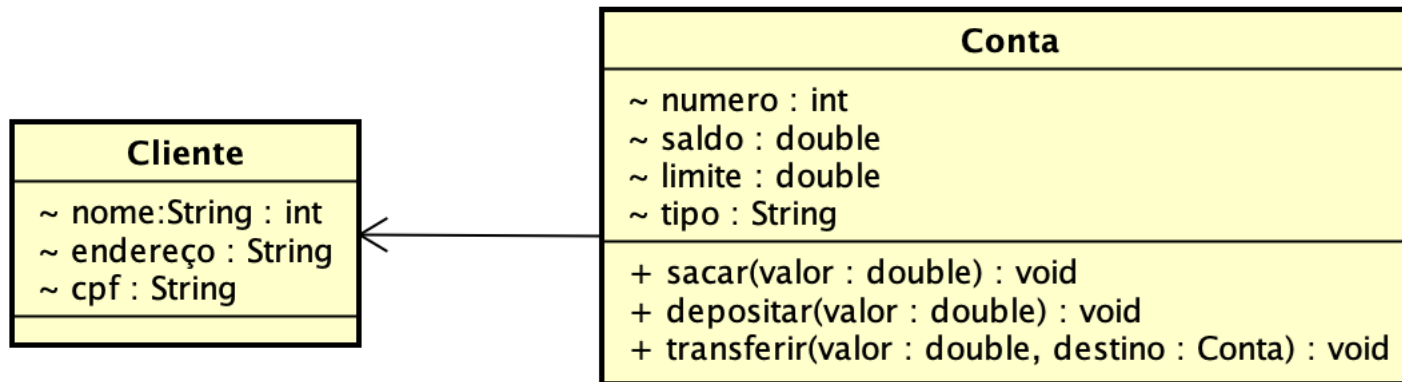
- Nome
- Endereço
- CPF
- Data de Nascimento
- Telefone
- Etc..

São atributos de Cliente e não de Conta

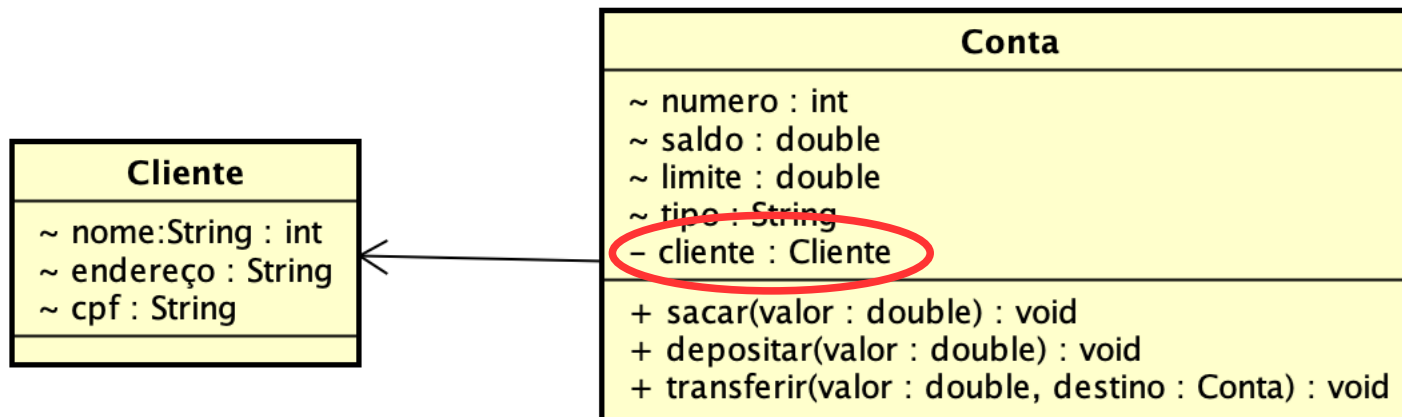
Um Cliente pode ter mais de uma Conta

Programação Orientada a Objetos

Modelo de Conta associada a um Cliente em UML

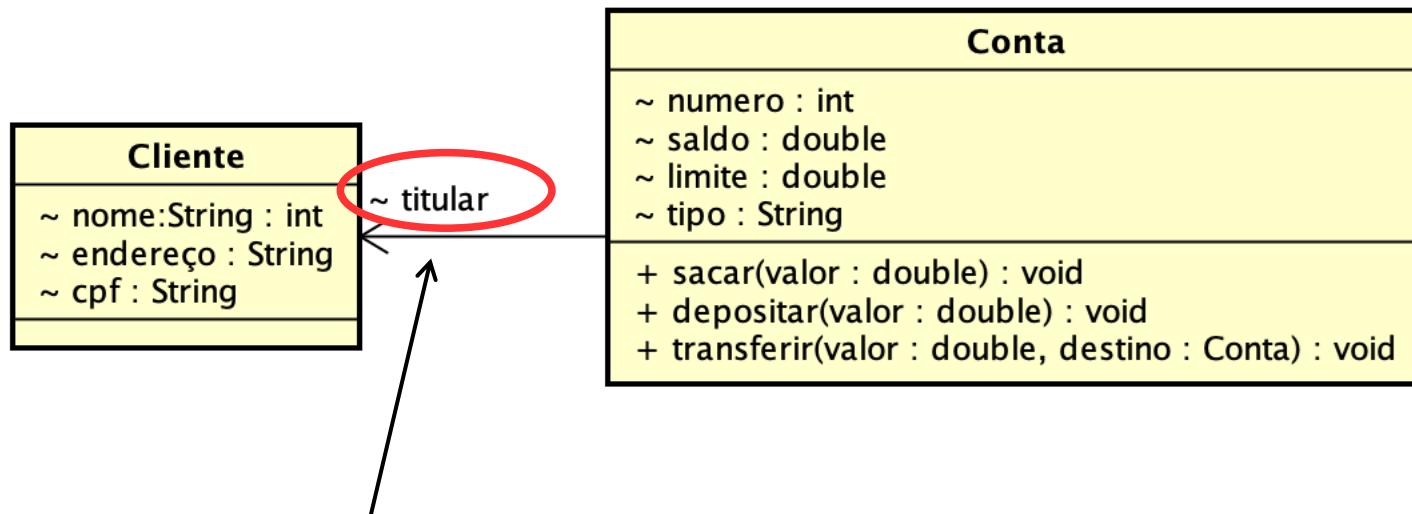


o mesmo que (não precisa representar o atributo no diagrama):



Programação Orientada a Objetos

Modelo de Conta associada a um Cliente em UML



Representação muito utilizada quando nome do atributo diferente do nome da classe

Programação Orientada a Objetos

Associando Classes (Versão 1)

```
class Cliente1 {  
    String nome;  
    String endereco;  
    String cpf; //...  
}  
  
class Conta1 {  
    int numero;  
    double saldo;  
    double limite;  
    String tipo;  
    Cliente1 titular;  
}
```

```
class Programa1 {  
    public static void main (String[] args) {  
        Conta1 minhaConta = new Conta1();  
  
        Cliente1 c = new Cliente1();  
        minhaConta.titular = c;  
  
        minhaConta.titular.nome = "Ana";  
  
        System.out.println("\nNome do titular da conta é " +  
            minhaConta.titular.nome);  
    }  
}
```

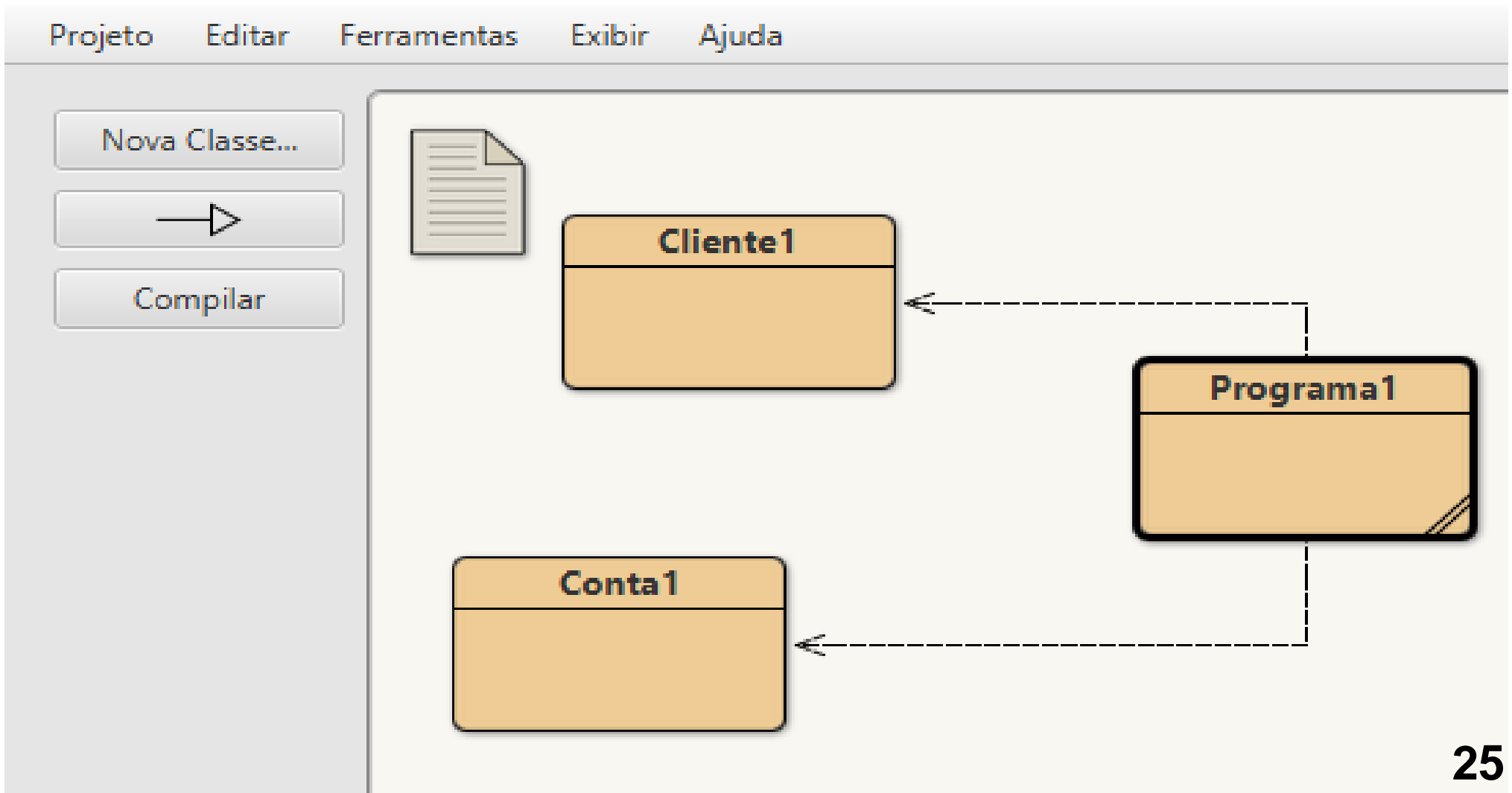
minhaConta agora tem uma referência para o Cliente correspondente a c

Forma de Acessar o Cliente

Programação Orientada a Objetos

Associando Classes (Versão 1) – Diagrama de Classe do BlueJ

 BlueJ:Associacaov1 ← **Nome do Projeto**



Programação Orientada a Objetos

Associando Classes (outra forma – versão 2)

```
class Cliente2 {  
    String nome;  
    String endereco;  
    String cpf; // ...  
}
```

```
class Conta2 {  
    int numero;  
    double saldo;  
    double limite;  
    String tipo;  
    Cliente2 titular = new Cliente2(); //...  
}
```

```
class Programa2 {  
    public static void main (String[] args) {  
        Conta2 minhaConta = new Conta2();  
        minhaConta.titular.nome = "Ana";  
  
        System.out.println("\nNome do titular da  
conta é " + minhaConta.titular.nome);  
    }  
}
```

Neste caso, toda conta quando criada terá um cliente associado a ela

Programação Orientada a Objetos

Associando Classes (outra forma – versão 2) – Diagrama de Classe do BlueJ

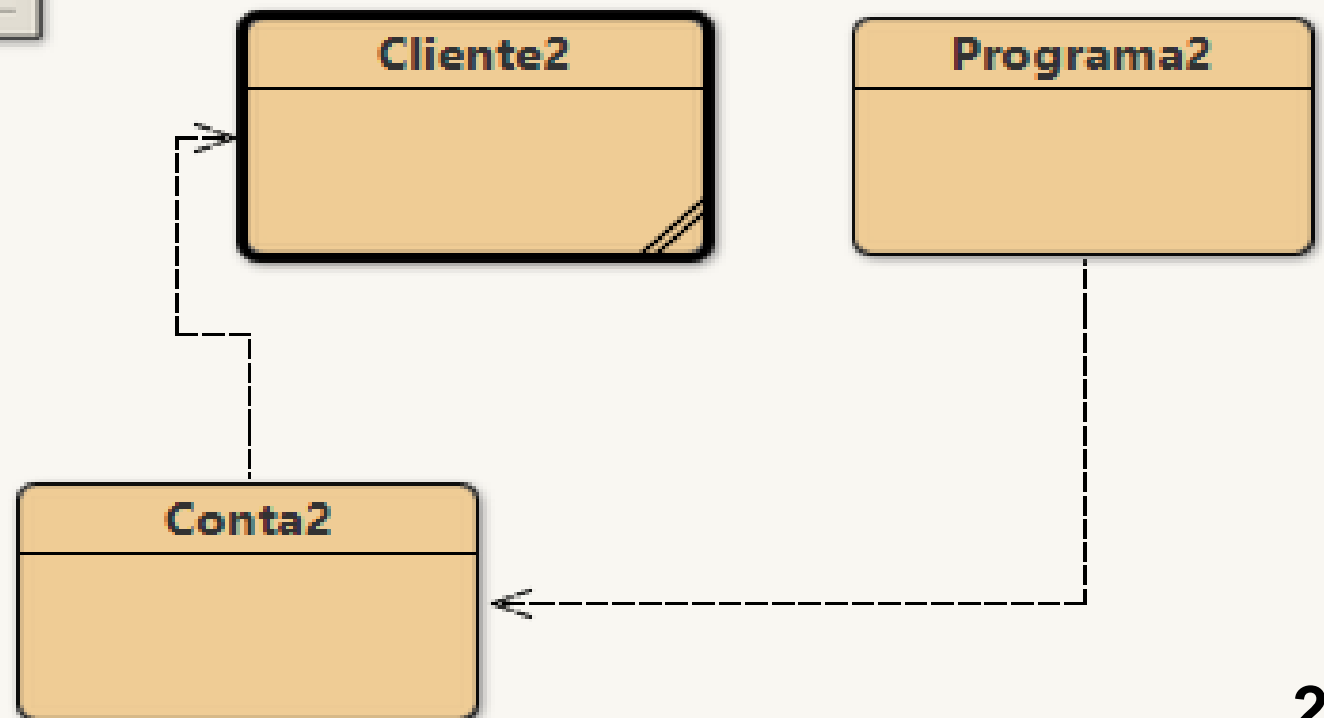
 BlueJ:Associacaov2 ← **Nome do Projeto**

Projeto Editar Ferramentas Exibir Ajuda

Nova Classe...

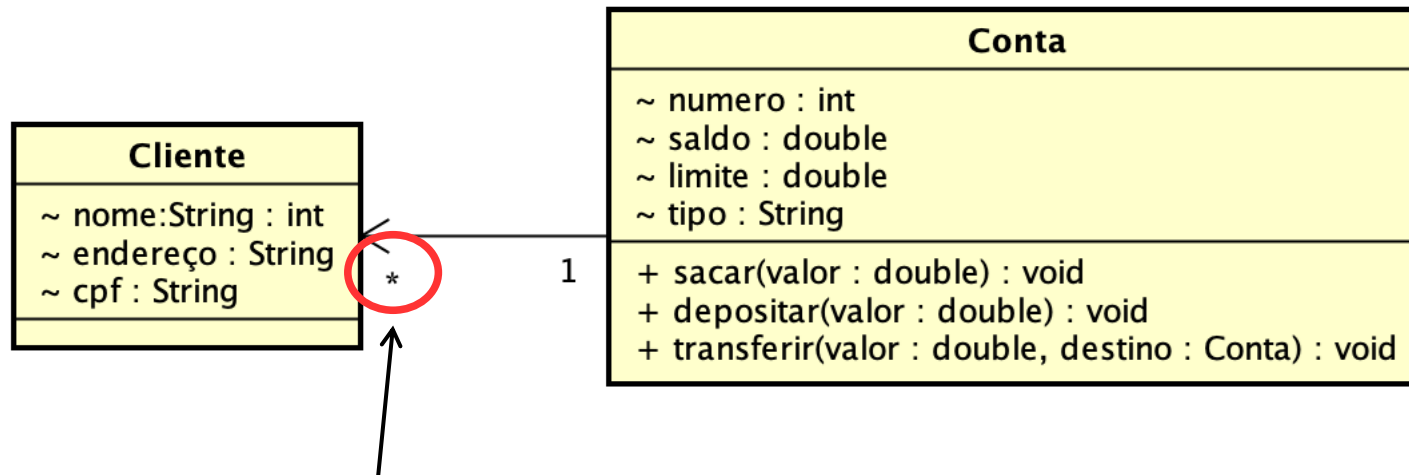


Compilar



Programação Orientada a Objetos

Conta associada a mais de um Cliente em UML



Uma conta pode ter vários Clientes

Programação Orientada a Objetos

Associando Classes (versão 3)

(Vários clientes em uma conta)

```
class Cliente3 {
    String nome;
    String endereco;
    String cpf; // ...
}

class Conta3 {
    int numero;
    double saldo;
    double limite;
    String tipo;
    Cliente3 clientes[] = new Cliente3[3];
}
```

```
class Programa3 {
    public static void main (String[] args) {
        Conta3 minhaConta = new Conta3();
        Cliente3 c = new Cliente3();
        minhaConta.clientes[0] = c;
        minhaConta.clientes[0].nome = "Ana";
        for (int i=1;i<3;i++) {
            minhaConta.clientes[i] = new Cliente3();
            minhaConta.clientes[i].nome = "Dependente" + (i);
        }
        System.out.println("\nNome do titular da conta é " +
            minhaConta.clientes[0].nome);
        System.out.println("\n Nome dos depedentes são:");
        for (int i=1;i<3;i++) {
            System.out.println("\n " +
                minhaConta.clientes[i].nome);
        }
    }
}
```

Programação Orientada a Objetos

Associando Classes (versão 3) – Diagrama de Classe do BlueJ

 BlueJ:Associacao3 ← **Nome do Projeto**

Projeto Editar Ferramentas Exibir Ajuda

Nova Classe...



Compilar

